AMENDMENTS TO THE SPECIFICATION

IN THE SPECIFICATION

[0010] A system and method are described for testing the random access memory of a computer system. The embodiments circumvent the time and efficiency problems inherent in testing by moving the testing procedure from a centralized testing system that must individually test each of the RAM's associated with the ASIC's to a memory testing engine (MTE) embedded on or coupled with the ~~slave-~~bus slave controller on each integrated circuit. This allows the testing to be performed on each RAM at once, reducing the time cost of testing each individual RAM. Additionally, by embedding the MTE in the ~~slave-~~bus slave controller, the amount of equipment needed to test the machinery can be reduced, increasing both fiscal and spatial efficiency. The efficiency is increased by shortening the path that the data has to travel, allowing the memory tests to be run concurrently, and freeing the processor to perform other functions. The speed, efficiency, and decentralized nature of the MTE will enhance field-testing of memory modules as well.

[0011] One embodiment of the method that may be used for testing is illustrated in the flowchart of Figure 1. The CPU would signal the memory test engine embedded in the ~~slave-~~bus slave controller to initiate a RAM test 100. The MTE selects a location in the RAM to test 110, and then writes a string of data to that location 120. The MTE then reads that string back from memory 130 and sees if it matches what was written 140. If the string does not match, the test is halted 150 and the CPU is informed of the error 160. If the string does match, and the RAM has been completely tested 170, the CPU is informed that the RAM is error free 180. If the RAM is not completely tested, a new location is selected 110 and the process is repeated.

[0012] An exemplary embodiment of the MTE 200 within a utility bus system is displayed in Figure 2. A central processing unit (CPU) daughter card 210 communicates over the protocol control information (PCI) bus 220. For one embodiment, the CPU daughter card contains a CPU 211, a chip set 212, and a memory storage device 213. For another embodiment, the PCI proprietary bus may have a PCI communications link (PCL) (PCI 0 message) 220 or Utility Bus (PCI 1 message) 221. The PCI bus accesses

a series RAM's to be tested via a translator 230, such as a Utopia Data Path 192 (UDP192) chip. The translator 230 would give the daughter card access to a master bus controller 240 and a series of ~~slave bus~~utility bus slave (UBS) controllers 250. In one embodiment, the translator 230 provides access to a master bus controller 240 and eight ~~slave bus~~UBS controllers 250. In an alternate embodiment, each ~~slave bus~~UBS controller 250 contains the functionality of a translator 230 and master bus controller 240, allowing the PCI bus 220 to communicate directly. Each ~~slave bus~~UBS controller 250 is coupled to a RAM or series of RAMs 260 via a memory controller 270. For one embodiment, the RAM 260 can be either a static RAM (SRAM) or a synchronous dynamic RAM (SDRAM). For a further embodiment, each ~~slave bus~~UBS controller 250 contains a MTE 200, which may be embedded within the ~~slave bus~~UBS controller 250, or separately coupled to the ~~slave bus~~UBS controller 250 via a register controller 280. Either arrangement allows for the MTE 200 to utilize the data, address, and control pathways used by the ~~slave bus~~UBS controller 250. Control of these pathways is passed between the MTE 200 and the ~~slave bus~~UBS controller 250 so that only one of these entities has control at a one time. For example, if data traffic is being passed to the memory modules by the ~~slave bus~~UBS controller, the MTE 200 cannot run a test function. In one embodiment, a bit register tracks whether the MTE 200 or the ~~slave bus~~UBS controller 250 has control of the pathways.

[0013] For one embodiment, the MTE 200 architecture would be constructed as illustrated in Figure 3. The processor configures the MTE 200 through write messages to the translator chip over one of the proprietary busses. The translator chip relays the message and translates the protocol between the proprietary buses to the MTE. These translated messages are seen by the MTE register interface logic 300 as register requests 301 with the register write 302 indication active. Associated with the register write request from the ~~Utility Bus Slave~~UBS controller 250, in one embodiment, is a register address 303 and register write data 304. The MTE register interface 300 decodes the register address 303 and writes the register write data 304 to the location indicated by the register address 303; for example an Instruction RAM (I-RAM) 305 word or a Constants RAM (C-RAM) 306 word whose RAMs have dedicated write interfaces with the Register Interface 300. Other writeable registers include a Control Register 307, and an Interrupt Enable register 308. The MTE register interface logic 300 acknowledges a

register-write request by asserting a register data acknowledgement 309 and thereby releasing the utility bus slave controller to accept a new bus message.

[0014] Using the methods described, the processor writes the MTE's micro-coded RAM test program 310 to the I-ram 305. The program needs to be loaded only once after power up. The MTE Enable Register 311 is written to disable the MTE by de-asserting the enable signal 312, which resets the program counter 319. The Enable Register 311 is then rewritten to enable the MTE, asserting the enable signal 312 to the ~~Utility Bus Slave~~UBS controller, which gives the MTE use of the memory interface 313. The MTE C-RAM 306 is written with the constants 314 necessary to tailor the test to the configuration of a particular RAM. Typically, the MTE Interrupt Mask Register 308 is configured to enable interrupts to the processor via the interrupt signal 315 for conditions indicated in the MTE Status Register 316, such as "Test completed successfully" or " Test failed". Finally, the Control Register 307 is written with the Target ID that indicates which RAM to be tested and a start bit 317 that triggers operation of the MTE Arithmetic Logic Unit (ALU) 318.

[0018] The memory interface 313 connects the MTE to the memory controller 270, which is shared with the UBS controller 250. The memory interface 313 accommodates communication between the MTE and different interfaces to memory ~~controllers~~ controller 270 (as used in a Utility Bus Slave) or memory ~~controllers~~ controller 430 (as used in a PCL bus configuration, see Figure 4). The ALU 318 generates a data pattern to write to memory and sends memory request 331, the write memory signal 324, the RAM target 332, the memory address 333, the write data 334, and the request length 335 to the memory interface 313. The data is accumulated in a memory interface data buffer. The memory interface sends a memory request 339, the memory write signal 340, a memory target 341, memory address 342, and memory length 343 to the memory controller 270/430. The memory controller 270/430 sees the request and responds by reading the data from the buffer in the memory interface 313. The data from the memory interface buffer 313 is read by sending the initial word address on the data word select signals 344. After a fixed latency, the memory controller 270/430 can sample the memory write data 345. The memory controllers advances the data word select 344 at the fastest rate that it can receive data. When the

memory controller 270/430 has transferred the number of words satisfying the length 343, the memory controller 270/430 asserts the memory done signal 328. The memory interface 313 signals done ~~328~~ to the instruction decoder 322 which allows it to continue to generate the next data pattern to be write, or to read the data back for comparison, as dictated by the program stored in the I-RAM 305.

[0019] Generally, the next instruction would generate a read request to the same memory target and location as seen by the assertion of the memory request 331, the de-assertion of the memory write signal 324, the same indication of the memory target 332, memory length 335, and memory address 333, as on the previous write request. The memory interface relays the message by asserting memory request 339 along with the de-assertion of the memory write signal 340, the memory target 341, the memory address 342, and the memory length 343 to the memory controller ~~255~~270/430. The memory controller ~~255~~ 270/430 signals memory read data valid 346, memory data word select 344, and the memory read data 347. The memory interface 313 passes the read data 348 and read data 349 valid to the word comparator 326 and the ALU 318. The word comparator 326 compares the read data 348 with the expected read word 337. If they match, a compare equal 350 is sent to the instruction decoder 322. The data word select incrementing while the data is returned until the requested read request length is satisfied, at which time the memory done signal 328 is asserted. The memory interface 313 signals the instruction decoder 322 that the data transfer is complete ~~328~~.

[0020] The instruction decoder generates the controls to compare the received memory data to the expected pattern. If the data does not match what is expected, an error bit is set in the interrupt to the MTE status register 316 and processing is terminated. If the corresponding mask bit is set in the interrupt mask register 308, an interrupt signal 315 is asserted to alert the processor. If the data does match what is expected, the address ~~in~~ is incremented, a data pattern is generated and a new memory write request is made to the memory interface 313. This process continues until an error is detected or the RAM has been completely tested. At that time, completion status can be written to status register 316 which, if the corresponding mask bit is set in the interrupt mask register 308, will cause the interrupt signal 315 to signal the processor.